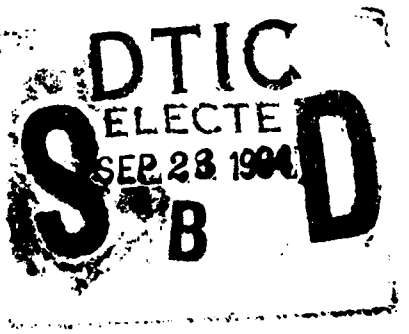
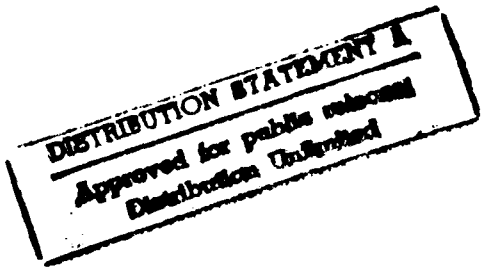
TASK: UU03
CDRL: 05156
February 1993

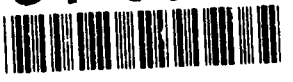
Reuse Library Framework User Tutorial

Informal Technical Data





STARS-UC-05156/018/00
February 1993



TASK: U03
CDRL: 05156
February 1993

INFORMAL TECHNICAL REPORT

For The

SOFTWARE TECHNOLOGY FOR ADAPTABLE, RELIABLE SYSTEMS (STARS)

RLF User Tutorial

STARS-UC-05156/018/00
February 1993

Data Type: A005, Informal Technical Data

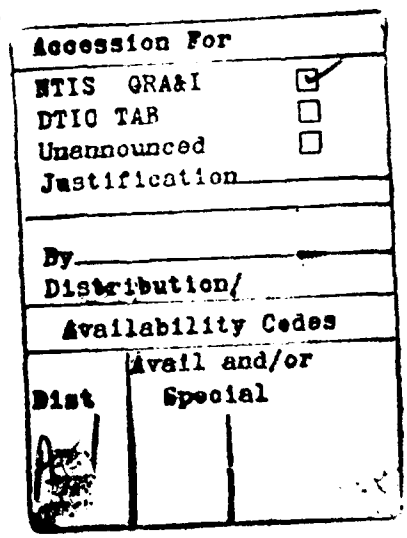
CONTRACT NO. F19628-88-D-0031
Delivery Order 0011

Prepared for:

Electronic Systems Center
Air Force Systems Command, USAF
Hanscom AFB, MA 01731-5000

Prepared by:

Paramax Systems Corporation
12010 Sunrise Valley Drive
Reston, VA 22091



TASK: U03
CDRL: 05156
February 1993

Data ID: STARS-UC-05156/018/00

Distribution Statement "A"
per DoD Directive 5230.24
Authorized for public release; Distribution is unlimited.

Developed by: Paramax Systems Corporation

This document, developed under the Software Technology for Adaptable, Reliable Systems (STARS) program, is approved for release under Distribution "A" of the Scientific and Technical Information Program Classification Scheme (DoD Directive 5230.24) unless otherwise indicated. Sponsored by the U.S. Defense Advanced Research Projects Agency (DARPA) under contract F19628-88-D-0031, the STARS program is supported by the military services, SEI, and MITRE, with the U.S. Air Force as the executive contracting agent.

TASK: U03
CDRL: 05156
February 1993

INFORMAL TECHNICAL REPORT
RLF User Tutorial

Principal Author(s):

Jim Solderitsch *Date*

Approvals:

Task Manager *Richard E. Creps* *Date*

(Signatures on File)

TASK: U03
CDRL: 05156
February 1993

INFORMAL TECHNICAL REPORT
RLF User Tutorial

Change Record:

Data ID	*Description of Change*	*Date*	*Approval*
STARS-UC-05156/018/00	Reissued: Minor changes to accompany RLF v4.1 release	February 1993	*on file*
STARS-UC-05156/008/00	Original Issue	November 1992	*on file*

REPORT DOCUMENTATION PAGE

Form Approved
OMB No 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
		Informal Technical Report

4. TITLE AND SUBTITLE

RLF User Tutorial

5. FUNDING NUMBERS

F19628-88-D-0031

6. AUTHOR(S)

Paramax Corporation

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Paramax Corporation
1210 Sunrise Valley Drive
Reston, VA 22090

8. PERFORMING ORGANIZATION REPORT NUMBER

STARS-UC-05156/018/00

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Department of the Air Force
Headquarter, Electronic Systems
Hanscom AFB, MA 01731-5000

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

05156

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION / AVAILABILITY STATEMENT

Distribution "A"

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

This is an initial version of an RLF end-user training package. This package complements training packages that cover RLF Modeling and Administration. The purpose of this package is to provide an orientation for new RLF users – those who are working with RLF applications in general and with the RLF Graphical Browser (GB) library interface in particular. This package does not assume that its audience is already familiar with the RLF and is meant to provide a general introduction to its use. The RLF User's Manual provides a more comprehensive view of RLF utilization than is presented here and persons requiring information beyond the perspective provided in this package should consult this manual. However, the audience is expected to have a general understanding of the UNIX operating system and operating within the X Window System. If X Window System experience is lacking, the prospective RLF user should identify a local source of expertise in this area.

14. SUBJECT TERMS

15. NUMBER OF PAGES
39

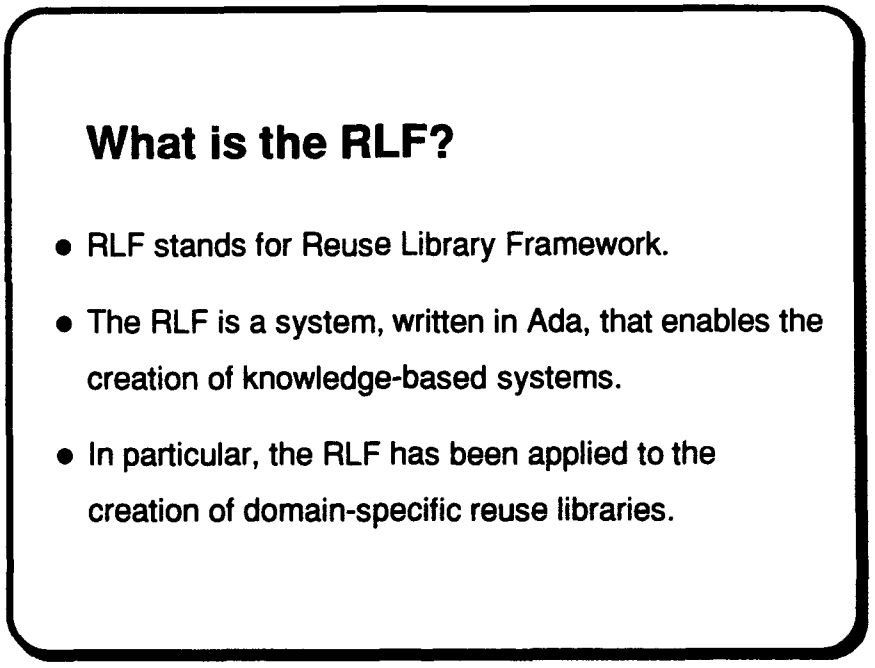
16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unlcassified	Unclassified	Unclassified	SAR

1 Introduction

The RLF is a knowledge-based system, developed by Paramax with support by the STARS program, whose primary application has been the design, implementation, and deployment of domain-specific reuse library systems. A reuse library supports software engineers by enabling them to quickly locate software assets (e.g. requirements, designs, code modules, test plans etc.) that can be of use in their construction of a software system.

Slide 1



A domain is an application area, typically the one of immediate relevance to the software engineer, and a domain model is a *machine representation* of information about the application-area and the library assets available for the application area. The model can contain general domain information along with data about the form, fit, and function of the available library assets.

This is an initial version of an RLF end-user training package. This package complements training packages that cover RLF Modeling and Administration. The purpose of this package is to provide an orientation for new RLF users – those who are working with RLF applications in general and with the RLF Graphical Browser (GB) library interface in particular. This package does not assume that its audience is already familiar with the RLF and is meant to provide a general introduction to its use. The RLF User's Manual provides a more comprehensive view of RLF utilization than is presented here and persons requiring information beyond the perspective provided in this package should consult this manual. However, the audience is expected to have a general understanding of the UNIX operating system and operating within the X Window System. If X Window System experience is lacking, the prospective RLF user should identify a local source of expertise in this area.

This orientation follows the organization of the RLF User's Manual which is part of the RLF

v4.1 release. The orientation follows the following general outline.

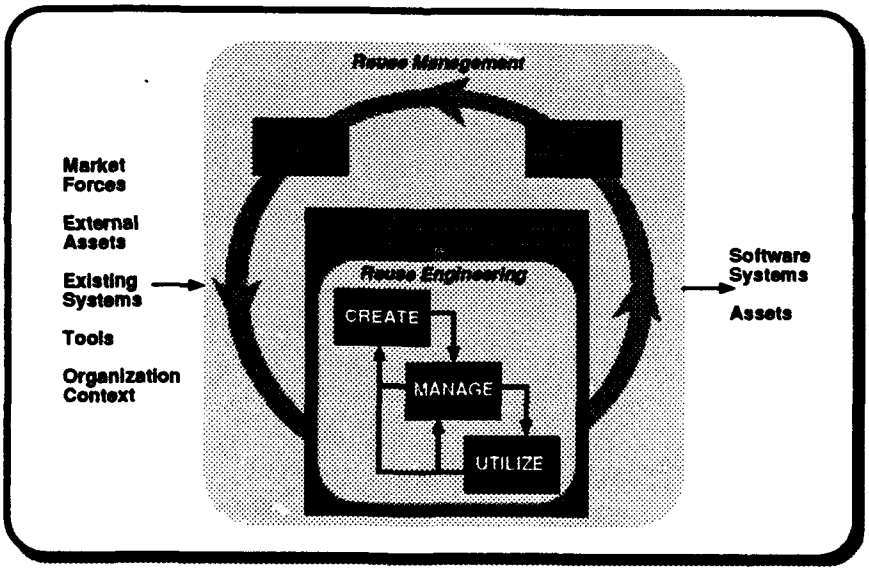
Slide 2

Orientation Outline

- Introduction
- RLF Background
- RLF GB Modes of Operation
- Context Sensitive Menus
- RLF Usage Tips
- Tailoring the RLF GB

The material in this orientation supports a portion of a domain-specific, reuse-based, process-driven approach to system engineering that is the cornerstone of the STARS program. This approach is embodied in the STARS Conceptual Framework for Reuse Processes (CFRP) shown on SLIDE 3.

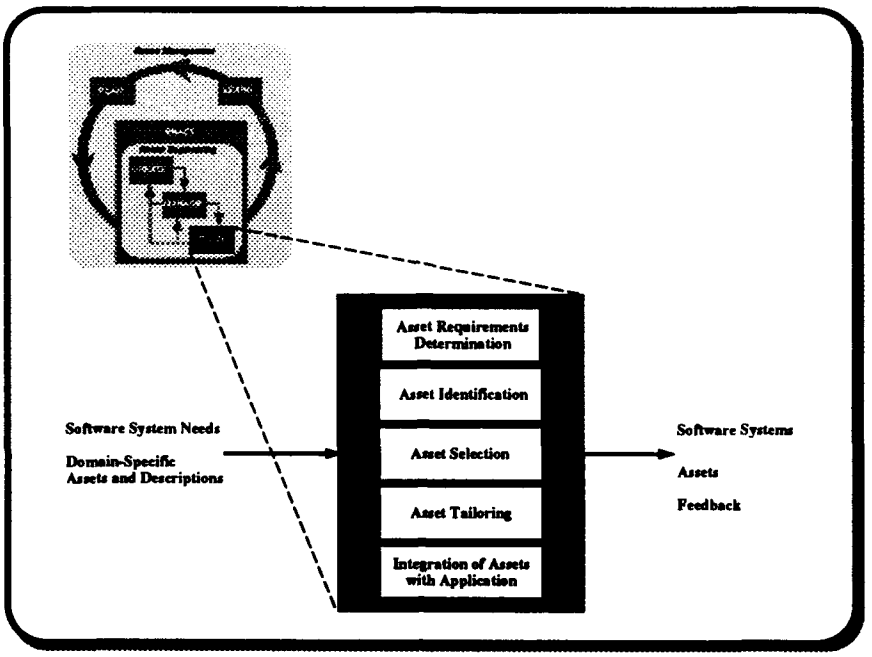
Slide 3



In particular, the RLF supports the creation of a reuse support structure and the management of assets stored within this system. An RLF end-user is most likely concerned with the utilization of assets that are modeled and stored within a library structure.

SLIDE 4 breaks out the Utilize activity on the CFRP. A well-developed and presented RLF model helps the user accomplish the first three of the tasks shown on the slide.

Slide 4



2 RLF Background

SLIDE 5 provides an outline of some basic RLF fundamentals that an RLF user should be familiar with. RLF's approach to managing a reuse library is based on the principle that a highly-structured reuse library will be easier to browse and understand. The structure of an RLF library is provided by a knowledge network which not only classifies the assets in the library in a hierarchy from general to most specific, but also describes the relationships between assets and the part they may play in the composition of a larger system.

Slide 5

RLF Fundamentals

- Domain Model approach
- RLF concepts
  - categories
  - objects
  - relationships
  - attributes
  - actions
- Inferencing

When an RLF library model which describes this knowledge network is constructed, the first step is to identify the area common to all the assets to be available in the reuse library. A domain-specific approach is summarized on SLIDE 6. Additional information on modeling a reuse library with the RLF is given in the **RLF Modeler's Manual** and the Modeling Orientation Package.

Slide 6

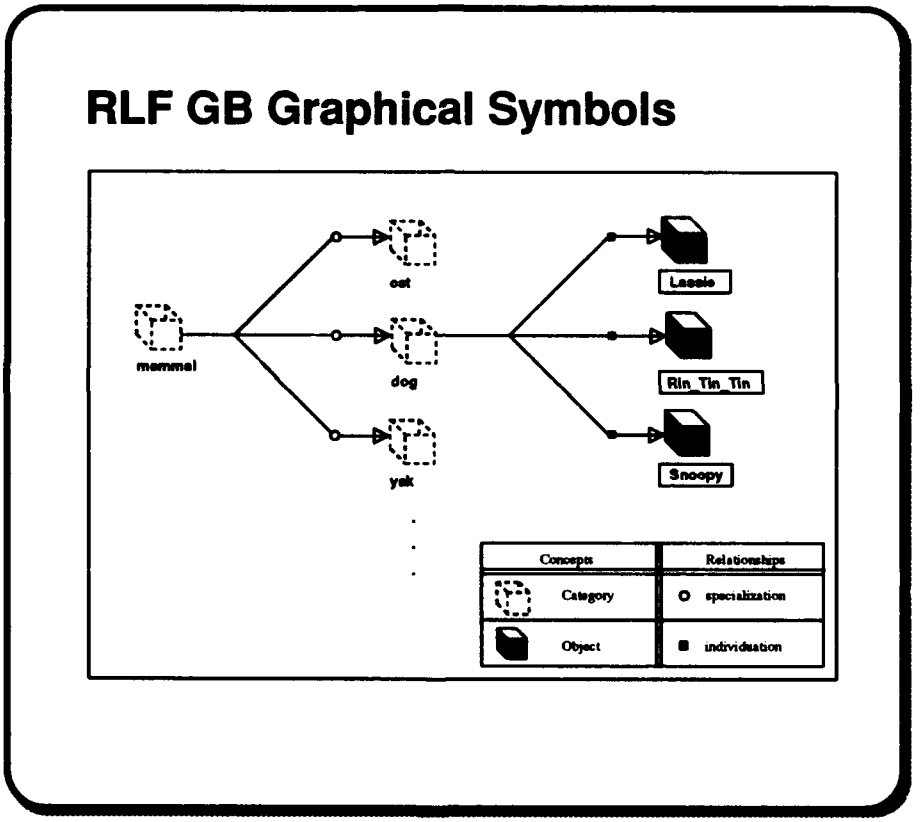
Domain Model approach

- Domain is an application area
- Domain Analysis produces knowledge about Domain
- Encoding knowledge leads to Domain Model
- RLF knowledge-base capabilities used to represent model
- Knowledge-Based approach leads to improved user effectiveness

The "domain model" is the final product of domain modeling. By capturing the domain model of the reusable assets in the library as an RLF knowledge network, the level of understanding of the assets in the library increases significantly. This in turn improves the chances that an asset extracted from the RLF library will be immediately useful to the library user. The key to effective reuse is to minimize the time taken to find and retrieve the asset to be reused, and to increase the chances that the asset can be reused without much alteration. The domain model approach ensures that there is enough information in the library structure to meet these goals.

Before discussing RLF concepts further, a brief look at the graphical notation used by the RLF GB for some of these concepts is shown on the next slide. The figure shows a category with three subcategories with one of the categories containing three objects.

Slide 7



SLIDE 8 summarizes the important features of model categories.

Slide 8

Categories

- general descriptions of a kind of thing
- classify what a thing is
- arranged in an inheritance hierarchy called a specialization hierarchy
- relationships inherited along specialization hierarchy
- most general category occurs at root of the model

Objects are all members of categories and are distinguished by the relationships they have as a result of their location in the specialization hierarchy. SLIDE 9 summarizes some key properties of objects.

Slide 9

Objects

- actual things instead of classifications of things
- associated with the most appropriate category (or categories) which describe them
- reusable assets are objects which are identified through attributes as well as category relationships which are "instantiated" for the asset
- objects are said to "individuate" the containing category

Categories and objects are distinguished primarily through the relationships and attributes which they possess. SLIDE 10 describes some basic properties of relationships in the RLF.

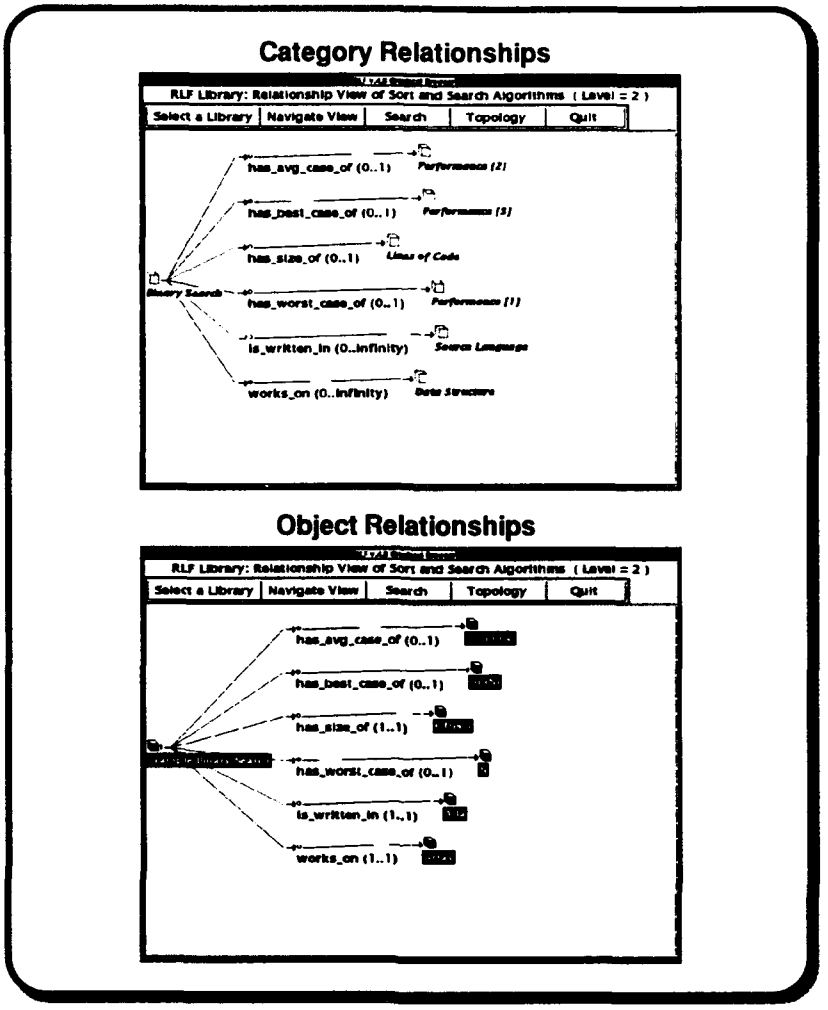
Slide 10

Relationships

- describe features of categories and objects
- express associations between different categories or objects
- are inherited by categories or objects below that category in the hierarchy at which they are defined
- have a name, an owner, a type and a cardinality
- can be restricted by type or by cardinality when inherited (but never generalized)
- object relationships "fill" corresponding relationships defined between corresponding categories

The next slide gives an actual RLF GB graphical view of relationships between categories and relationships between objects. In the second figure, an object in the `Binary Search` class is shown along with the filled relationships for each of the relationships defined for that class. Each relationship is instantiated with a target object which is a member of the category corresponding to the type of the relationship. The example also shows that cardinalities of relationships can also be restricted when they are filled.

Slide 11



RLF attributes are used to annotate categories and objects with static information that provides additional data about them. Attributes are the means by which actual file contents are attached to reusable assets which are modeled as objects in an RLF model. SLIDE 12 illustrates some features of attributes.

Slide 12

Attributes

- can be integers, strings of characters, or files
- have names so they can be referenced and used by RLF *actions*
- file attributes can be viewed, extracted, or otherwise manipulated
- attributes are not inherited -- they must defined at each category or object where they are useful

Actions let the user process categories and objects and permit access to system and library resources within a context appropriate to specific categories and objects. The context for these actions is provided by RLF attributes. Actions are summarized in SLIDE 13. Action definition is covered in both the Administrator's and Modeler's orientation packages.

Slide 13

Actions

- provide library users with appropriate system and library services
- basis for asset viewing and extraction
- defined within model definition language -- sample Action sub-model provided in RLF distribution
- implemented as system calls, or internal Ada procedures
- inherited like relationships
- have names, an action category, a list of action targets, and a list of action agents
- targets reference attributes which provide input for the action
- agents reference attributes which can affect how the action is performed
- can be privileged which means they are unavailable through the RLF GB (restricted to administrative use)
- can be restricted at subcategories or lower level objects

Finally, RLF models can make use of a special, built-in action called inferencing. This capability has been used primarily to direct a user browsing an RLF model and to provide that user with advice about the model in a context-dependent, goal-oriented fashion. The user's responses to inferencing-induced questions lead the user to categories and objects within the model that are expected to meet user needs as these are elicited during the inferencing process. An inferencing action is labeled "Advice" on an RLF Pop-Up menu when a selected category or object is equipped with such an inferencer option. SLIDE 14 describes some inferencing features.

Slide 14

Library Inferencers

- defined in special-purpose RLF language
- attached only to modeler-specified categories or objects
- ask the user questions about the user's needs and intentions
- based on responses, make deductions about model locations and contents that are of interest to the user
- can focus user to specific category or object
- are distributed over a model and can invoke and exchange information with each other

3 RLF GB Modes of Operation

The "feel" of an RLF application is dependent on the user interface of that application and on the capabilities of the workstation or terminal being used to interact with the application. The information in this training package is primarily geared to Sun workstation-equipped users who are running the X Window System, Release 4 (or greater).

Slide 15

Using the RLF

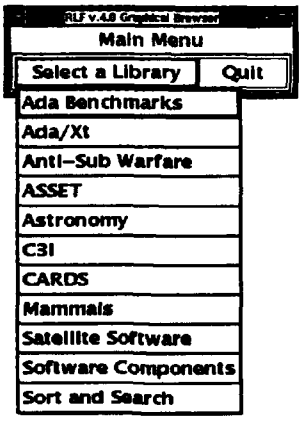
- RLF applications separate their user interface from their underlying knowledge bases.
- Both textual and graphical interfaces are provided to support users with different interface capabilities.
- The RLF graphical library interface is called the RLF GB -- it requires an X11R4 Window System end user environment.
- The next group of slides will survey using this interface.

A library application end-user invokes the application from an X windows terminal shell and is greeted by a menu panel that is used to select the library model to be accessed.

Slide 16

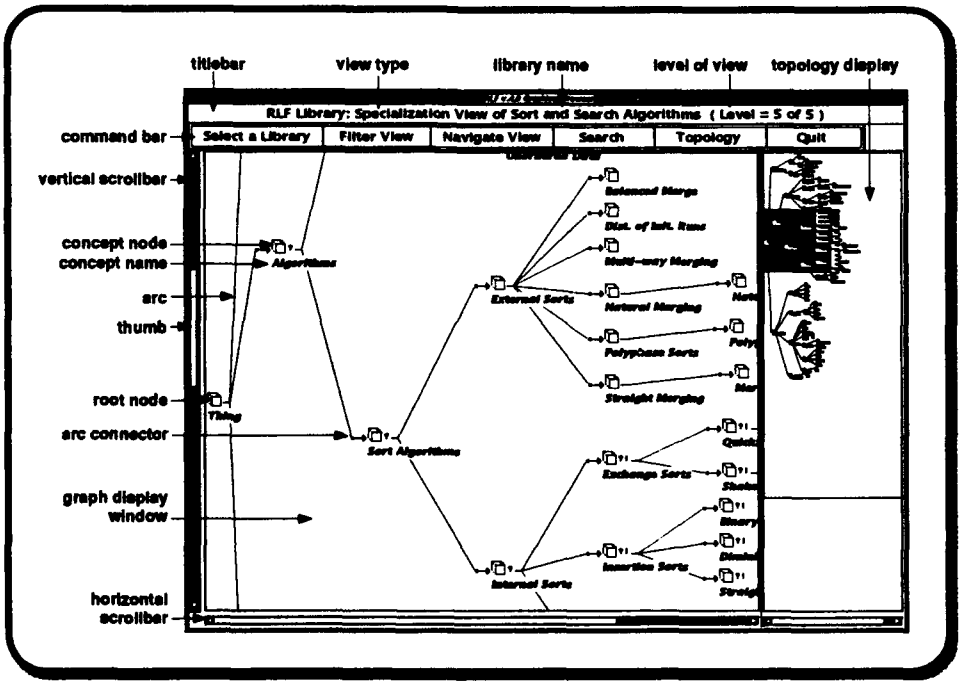
First Level RLF User View

After selecting the library through a pop-up menu, the RLF GB User sees the screen shown on the next slide.



SLIDE 17 shows an RLF screen which gives a graphical layout of the basic underlying network model for the library.

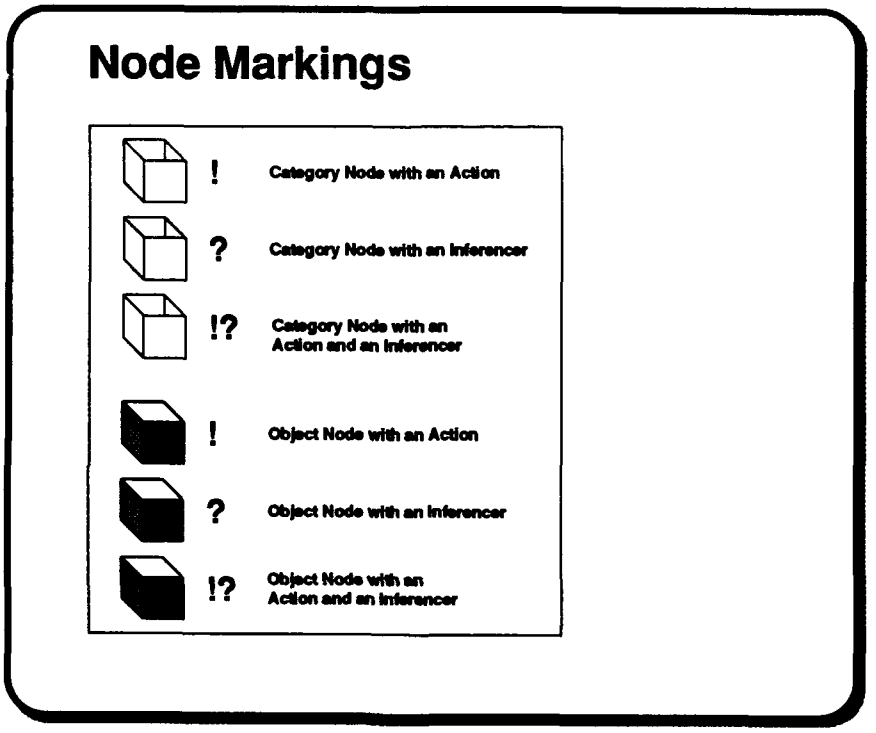
Slide 17



The main user options for using this graphical view are contained in pull-down menus at the top of the screen (e.g. Filter View, Navigate view etc.). The next few slides survey the options available from these menus.

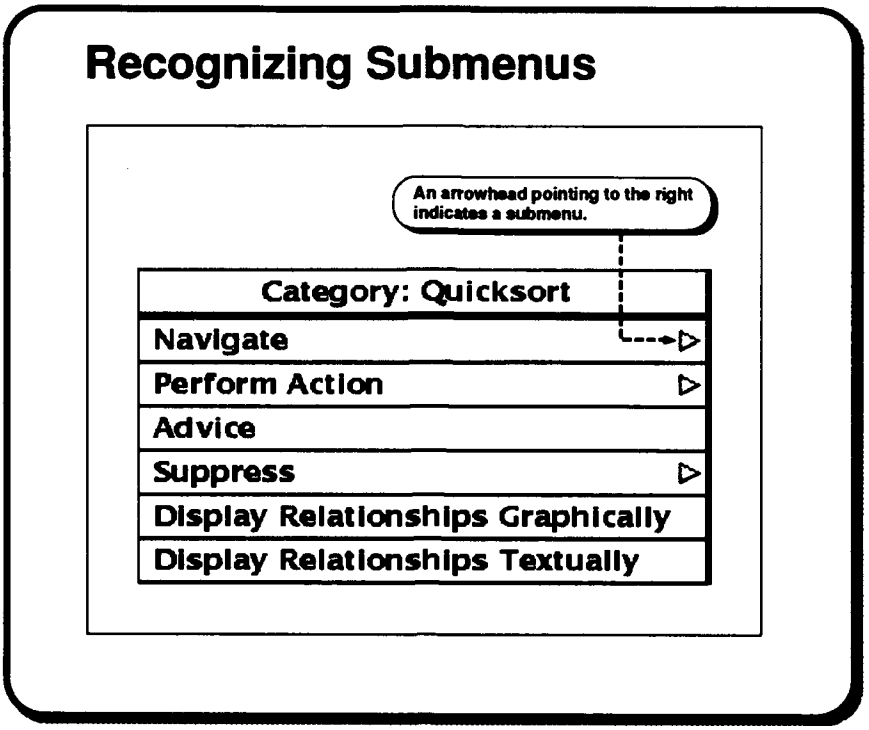
To provide more immediate indication to the user about the resources available at a given model category or object, the basic box icon can be marked with either an exclamation point or a question mark. The meanings of these marks is summarized on the following slide.

Slide 18



Many of the menus include cascading submenus. The presence of these submenus is indicated by a rightward-pointing triangle as shown on the next slide.

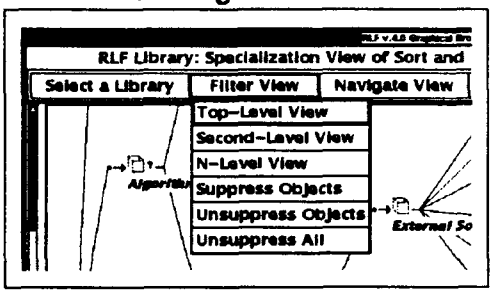
Slide 19



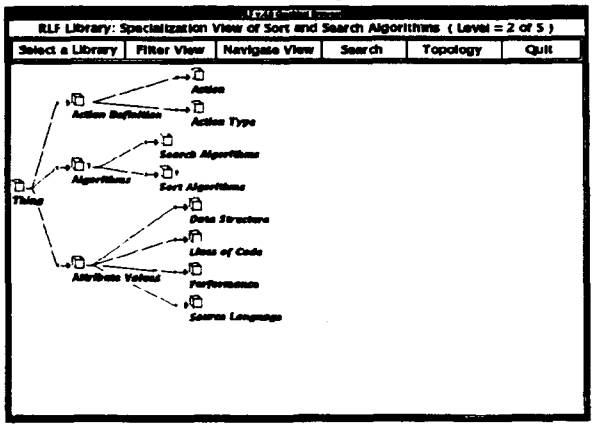
Drop-Down menus emerge whenever one of the commands in the menu bar is selected with the mouse. For example, selecting the **Filter View** command allows the user to tailor the current view to restrict (or unrestrict) the information that is displayed there. If the user selects **N-Level View**, then a Pop-Up dialog box appears in which the user types how many levels of the current model should be displayed. The next slide shows the appearance of the Drop-Down menu and the result of selecting a 2-level display.

Slide 20

Selecting Filter View



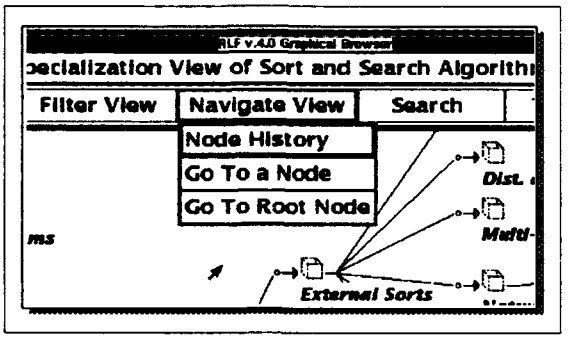
2-Level Display



SLIDE 21 shows the options available under the **Navigate View** command. The last option scrolls the view so that the root node of the model is clearly visible. The first node provides access to a list of nodes which the user has recently visited in the current session. These nodes are marked with an **S** if the user has visited the node in the usual browser view or an **R** if the user has visited the node in a graphical relationship view. Both relationship and specialization views may be on a user's display simultaneously (although one may be hiding (parts of) the other). The effect of choosing **Node History** is shown on the next slide.

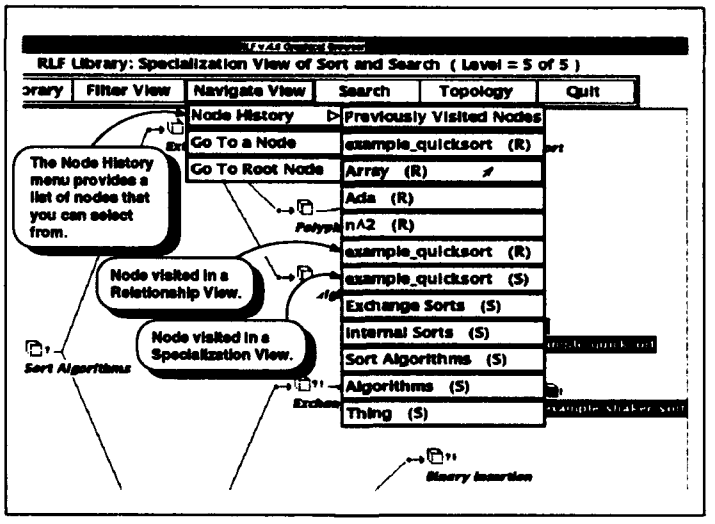
Slide 21

Simple Navigation



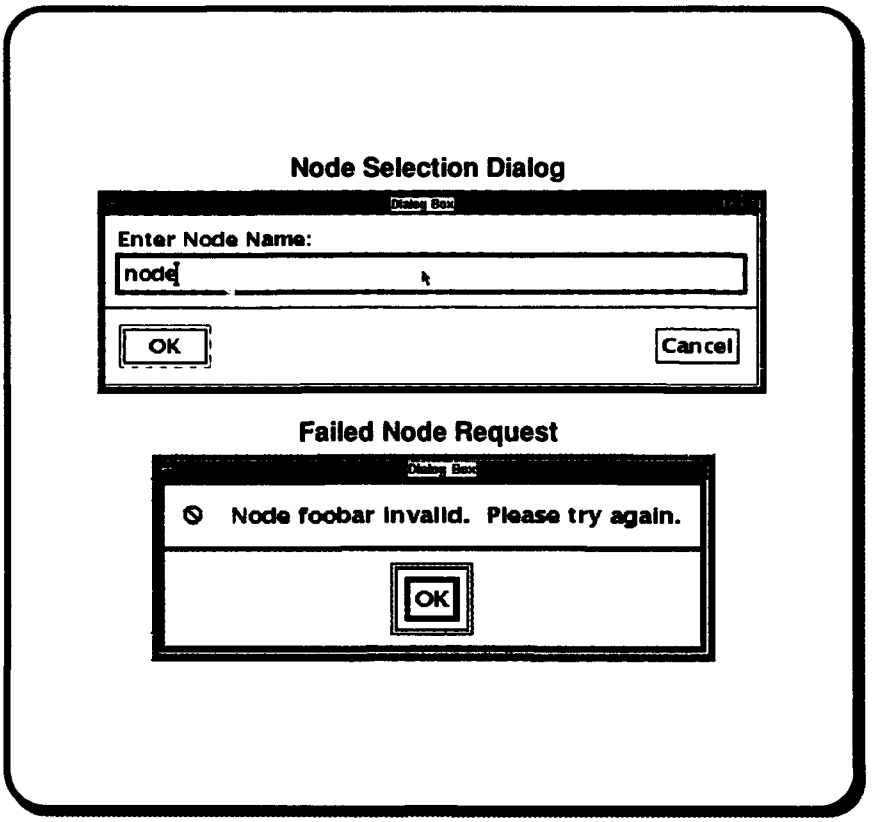
Slide 22

Node History Menu



SLIDE 23 shows how to locate a particular library category or object by name in the library model. A dialog box appears and the user types the desired name. If a matching node is found, the display is centered there. If not an Alert box appears prompting the user to try again.

Slide 23

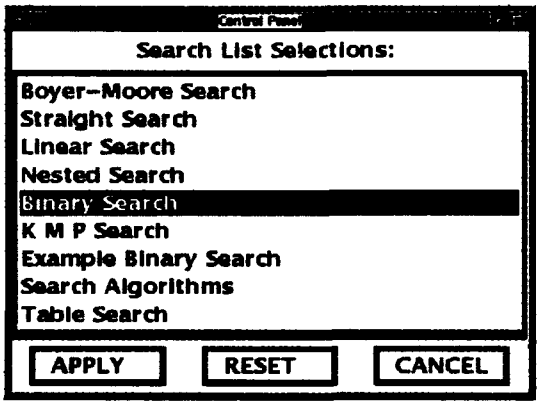


If a user wishes to locate categories or objects within the library, the search command is available from the command bar. If the user selects this option, a dialog box similar to the one shown in SLIDE 23 appears. The user enters at least three characters which are likely to be contained in the name of the node being searched for. After closing the dialog box, a list window similar to that shown in SLIDE 24 appears which holds the names of nodes containing the entered string. To visit a node from the list, the user selects the name and presses the **APPLY** button with the mouse. To erase the current selection, the user hits the **RESET** button. To cancel the search, the user hits the **CANCEL** button. If no node names contain the entered substring, a failure alert similar to that shown in SLIDE 23 appears.

Slide 24

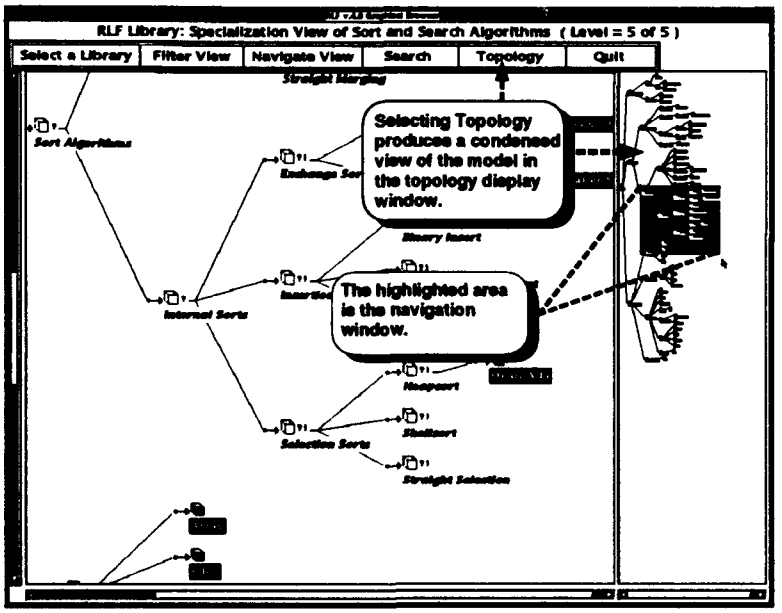
Search by Substring

In this example, a user is searching for nodes containing 'sea'. The list holds all nodes containing this string



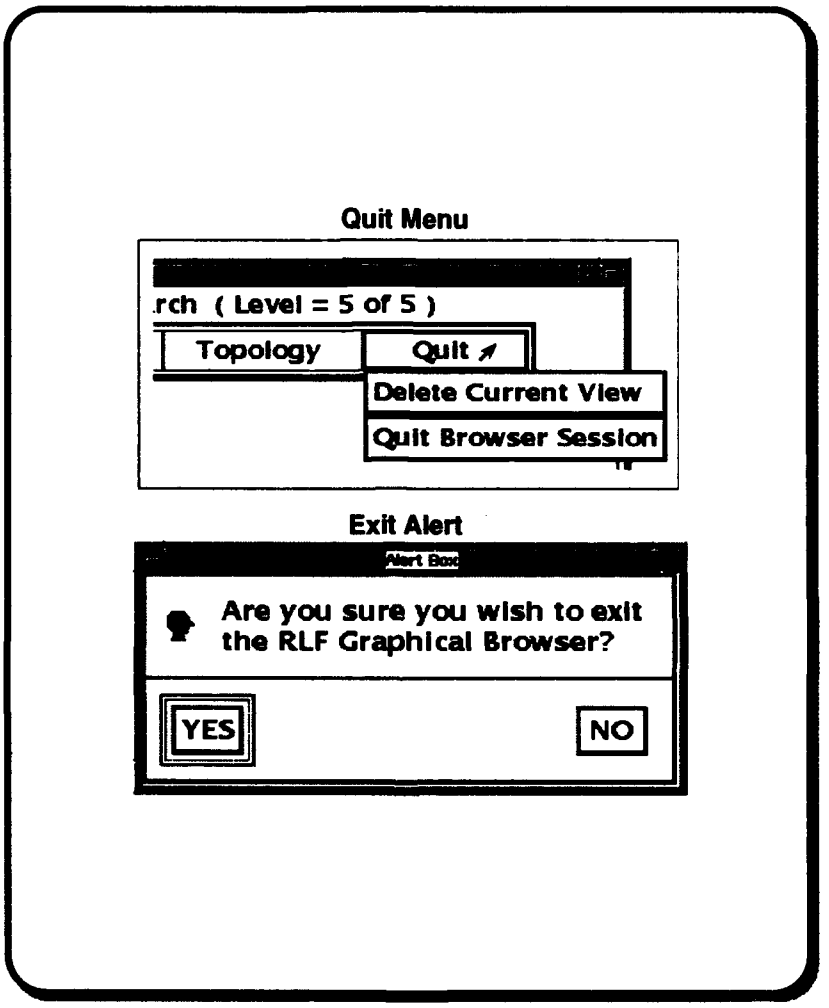
In a large model, it is helpful to see in a context surrounding the portion of the model that is currently within the main graphical view. The topology menu option provides a birds-eye view of a larger portion of the model (see SLIDE 25). The user can make large-scale moves in the model by pressing the mouse button on a location within the condensed view.

Slide 25



Finally, the user can select the **Quit** command. The options available here are to quit the current view (leaving any other views open) or to quit the current session entirely. If the latter choice is made (or if there is only one view currently open), the user is greeted with a confirmation dialog to see if the exit operation should be completed. This capability is shown on the next slide.

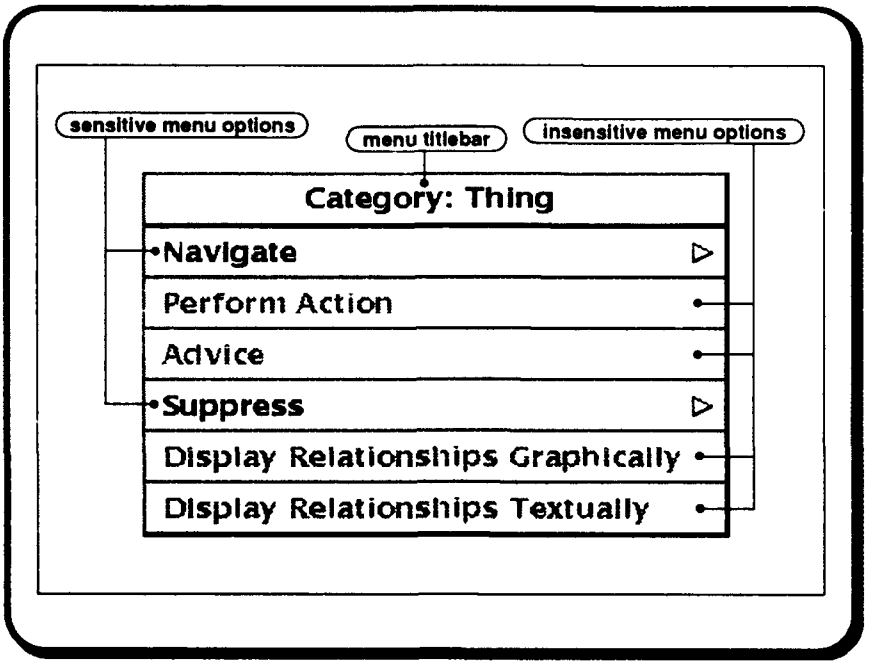
Slide 26



4 Context Sensitive Menus

Given a particular model location (class or object node) within a model, the user is able to navigate further through the use of pop-up menu selections available by pressing a mouse button when the mouse cursor is on top of that location. The typical list of options is shown on SLIDE 27. As the figure labels show on the on the slide, only those options appropriate for the node (category or object) are enabled. The rest are "greyed out" indicating that they are inappropriate for the current node. In addition, for some options, the availability of sub-menu options, and the content of these submenus, is also context sensitive. The series of slides in this section surveys some of the context sensitive options and their effects. Additional information about these options is found in the User's Manual.

Slide 27



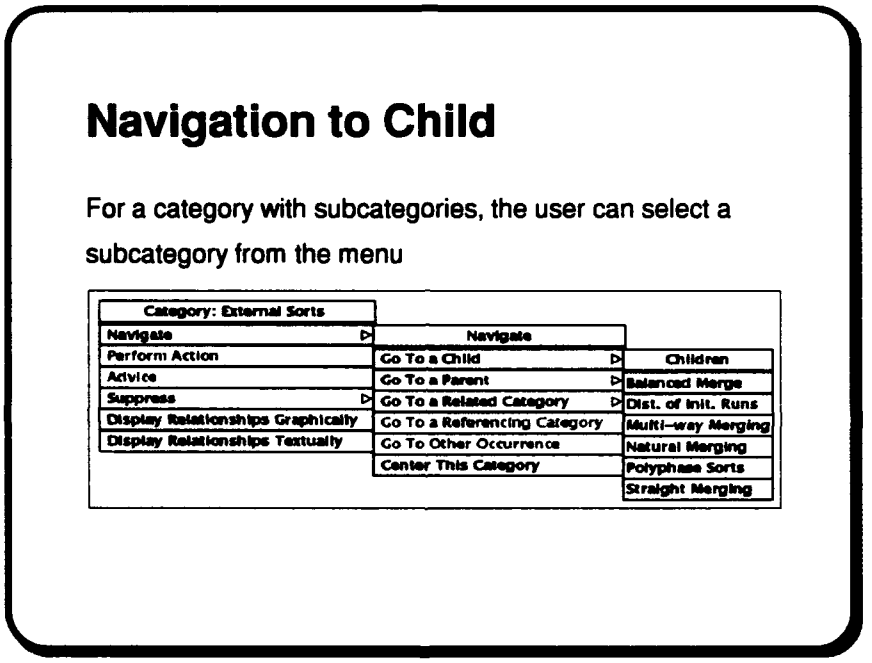
Once the user has made a first level selection from the navigation option, pop-up submenu selections are made available that are tailored to the kind of graph node at which the user is currently focused. The sub-menu options are also tailored to the kind of view that the user is browsing. In particular, the navigation sub-menu items are quite different within a relationship view compared to the specialization view. The complete subset of navigation options for the relationship view is as follows (only some of these are available at a specific node):

- **Go To a Referencing Category** ▷ (if the node is a category)
- **Go To a Referencing Object** ▷ (if the node is an object)
- **Go To a Referencing Node** ▷ (if the node is a category/object)
- **Go To a Related Category** (if the node is a category)
- **Go To a Related Object** (if the node is an object)
- **Go To a Related Node** (if the node is a category/object)
- **Go To Target** ▷ (if the node is a relation)
- **Go To Source** ▷ (if the node is a relation)
- **Go To Other Occurrence** ▷ (if the node has multiple occurrences)
- **Center This Relation** (if the node is a relation)
- **Center This Category** (if the node is a category)

- **Center This Object** (if the node is an object)
- **Center This Category in Specialization View** ▷ (if the node is a category)
- **Center This Object in Specialization View** ▷ (if the node is an object)

The next few slides explore navigation options within a specialization view. For example, if the node corresponds to a library category, the user can search among the parents, children or other related categories. SLIDE 28, SLIDE 29, and SLIDE 30 show some of the possible cascaded menus for navigation at a library category: children, parents and related categories. For a node with no children, the **Go To a Child** will be greyed out.

Slide 28

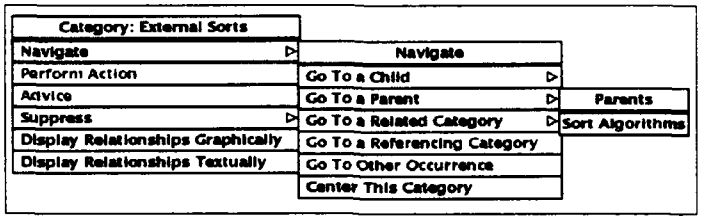


In the case of nodes with multiple parents, all parent node names will be accessible from the sub-menu.

Slide 29

Navigation to Parent

Every category except the root has at least one parent; the user can select a parent from the sub-menu.

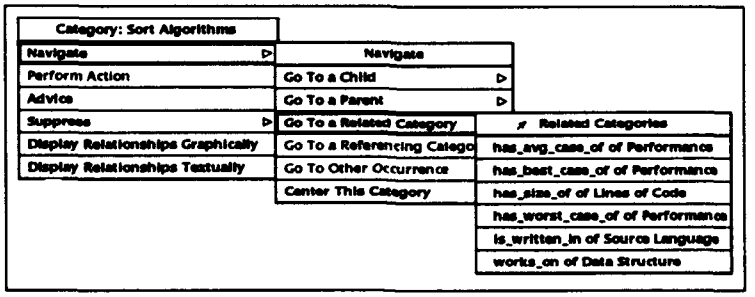


Related categories are those for which the current node is the source of a named RLF relationship that targets that category. The **Go To a Related Category** option will access all relationships that the current category possesses, including those inherited from its parents. These are listed by name, and include the target category for the relationship. Selecting one of them causes the user to move to the target category of the selected relationship.

Slide 30

Navigation to Related Categories

Category relationships can also be quickly selected from those available.

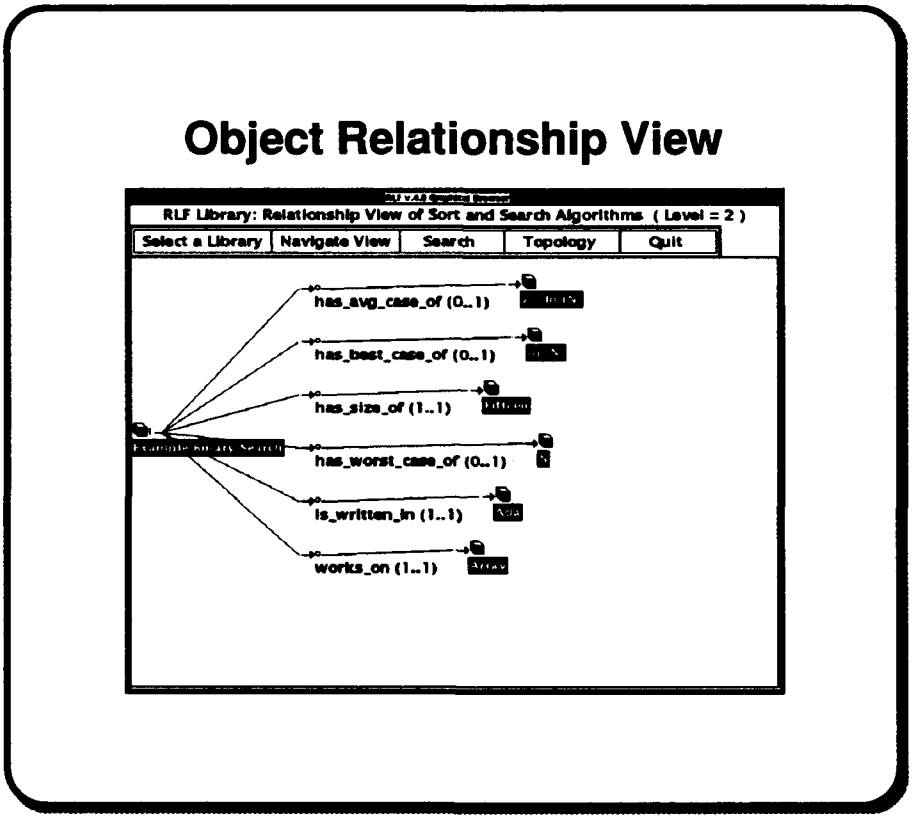


If the selected category does not have any relational attributes, then the **[Go To a Related Category]** option is greyed out in the navigate submenu. When the current node is an

object, the name of the sub-menu changes to **Go To a Related Category/Object** so that if a target object is available for a filled relationship, the user can navigate to that object. If the object relationship remains unfilled, the target category is available from the sub-menu.

Relationship views prevent a different set of navigation choices. When a node has relationships, the navigation option **Display Relationships Graphically** in the specialization view will be available. If this option is selected, a new view is created which can be navigated manually or through its own set of Pop-Up menu options. SLIDE 31 shows a Relationship View for the `Example Binary Search` object.

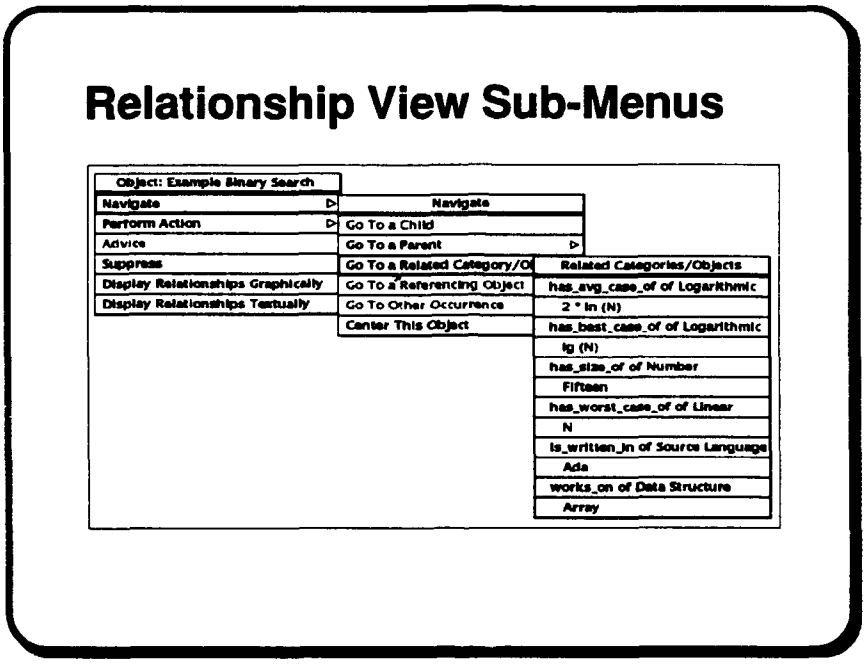
Slide 31



Within such a view, Pop-Up navigation options are tailored to this view and the node within the view that is currently selected. The user may also use Pop-Up navigation options directly on relationships where the user may select either **Go To Target** and **Go To Source** as options. Otherwise, the list of available options in the Navigation view is similar to those available in the specialization view.

An example is given in SLIDE 32 where the relationships on display graphically in SLIDE 31 are made available as menu options that the user can select to shift attention either to the relationship itself, or a target object or category referenced by that relationship.

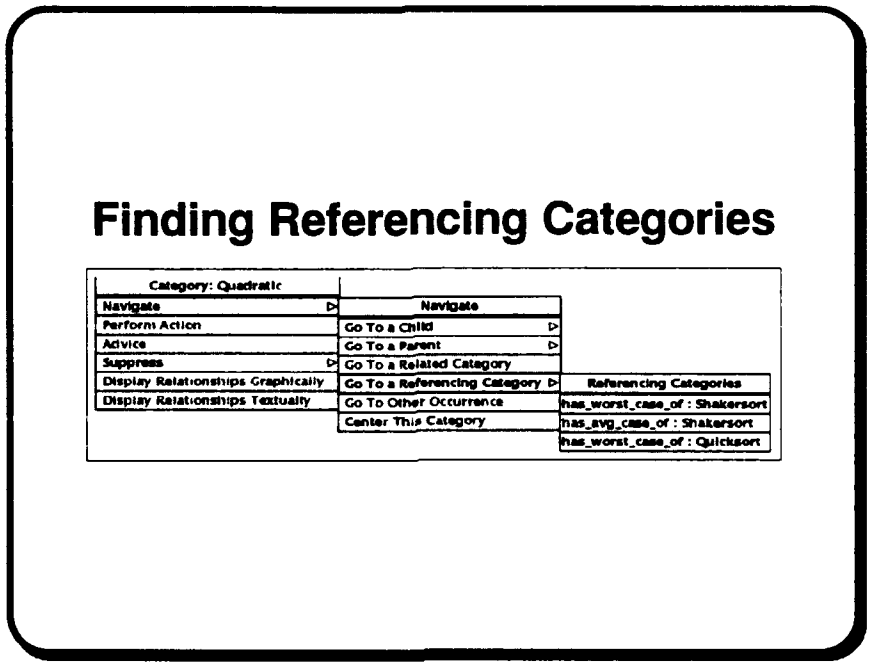
Slide 32



Help for traversing relationships in the opposite direction, from target back to the source, is also available in either specialization or relationship views. If a node is the target of any relationship a **Go To a Referencing ...** sub-menu item will be available for that node, and by selecting from the list of category and/or object names, the user can walk the hierarchy backward.

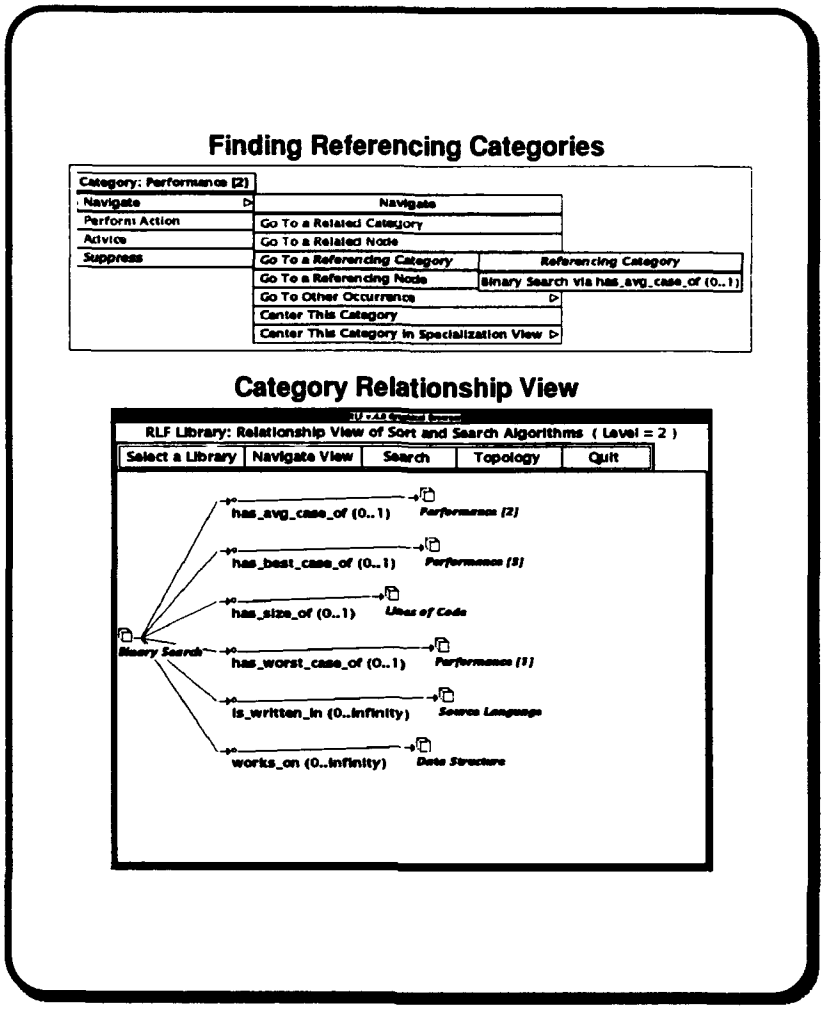
SLIDE 33 shows a cascaded menu for a category which is the target of several different relationships. These relationships and their sources are shown along the right edge of the slide. For example, the relationship `has_worst_case` references the category `Quadratic` and originates at `Quicksort`. Selecting this menu option selects and centers the display at `Quicksort`.

Slide 33



The previous slide shows inverse traversal within the specialization view. The next slide shows an analogous menu selection for the category `Performance` within a relationship view (which is shown at the bottom of the slide).

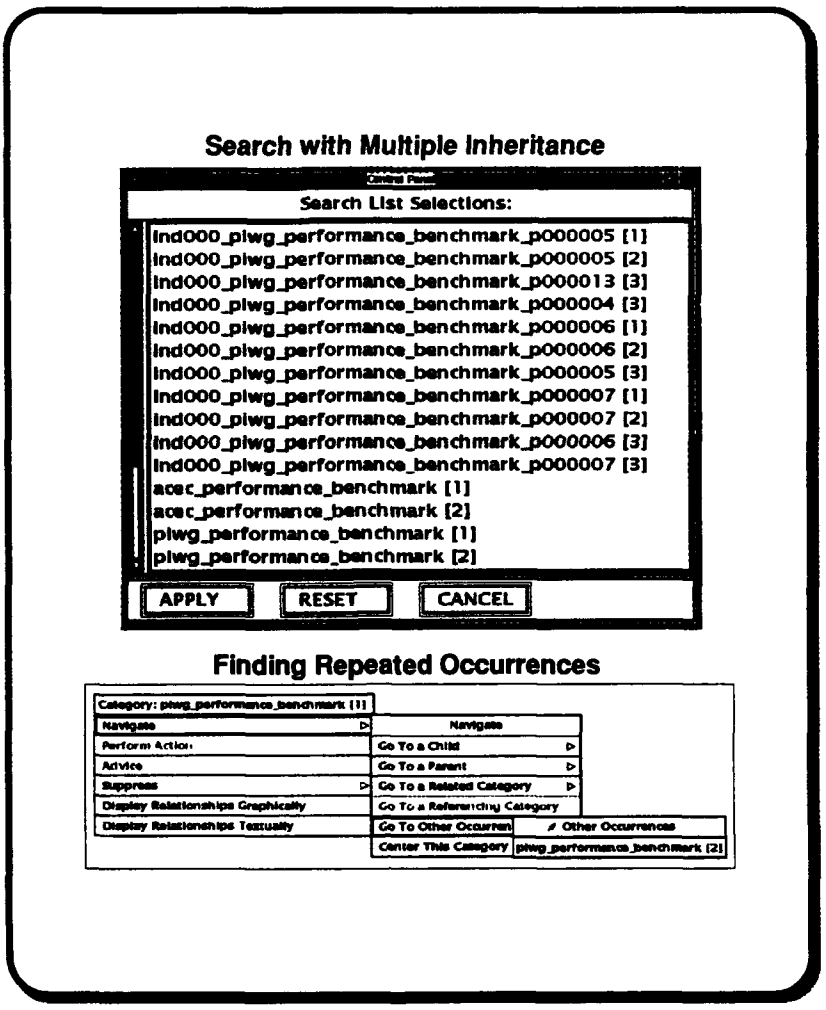
Slide 34



Similar backward traversals are possible for object nodes in either view. For example, having located the object `Ada` in the model and assuming that there exists a `written_in` relationship, this inverse traversal can be used to locate all objects which target `Ada` with the `written_in` relationship; i.e., all objects written in Ada.

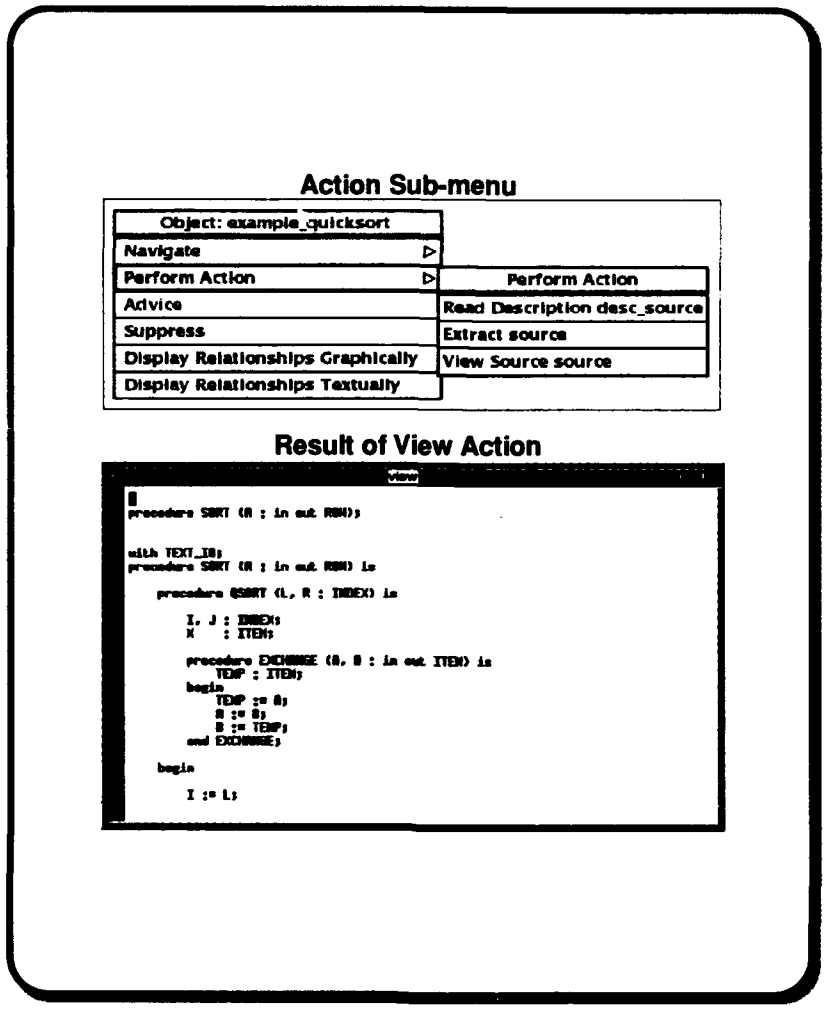
The numbers in parentheses in the previous slide inform the user that there is really only one node in the model with name `Performance`, but that since the display is laid out like a tree, the node is replicated to create an aesthetically pleasing appearance. In a large model, there may be many replicated occurrences of nodes such as these, and the user may wish to navigate among them. The sub-menu option **Go To Other Occurrences** makes this navigation easy. One possible scenario where this capability would be useful is when via a search command, a list of nodes is generated and some of these are repetitions of the same node. These replications will occur when the model includes multiple inheritance. Having selected one of the occurrences, the user may wish to see them all to become familiar with the multiple parents. 35 shows a list of nodes resulting from a search and then the use of the **Go To Other Occurrences** option to move among them.

Slide 35



Another feature available from the Pop-Up node menu for selected nodes is **Perform Action**. The actions available depend on the node and on the library model's definitions for the node (or its parents since actions are inherited). The action sub-menu gives the name of the action, and if available, the name of the attribute which is used as the "target" of the action. The target typically provides input for the action. SLIDE 36 shows a cascaded menu for a node with actions and a screen snapshot showing the effect of selecting the View action.

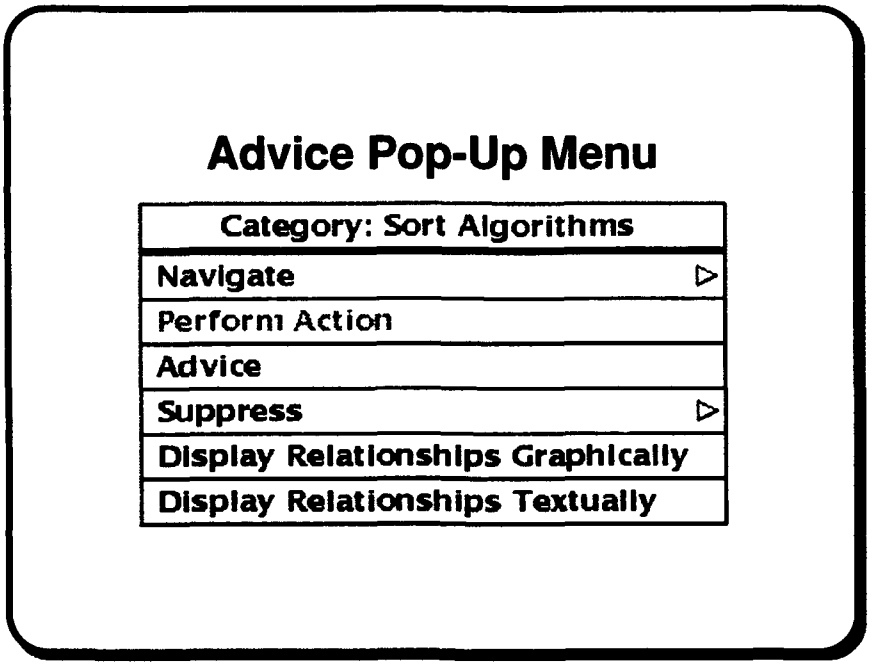
Slide 36



The last context-sensitive node menu option to consider is that of **Advice**. Some nodes in the model provide access to domain model information of a more heuristic and procedural nature, and by selecting the advice option, the user can receive advice about categories and objects which are contained in the model.

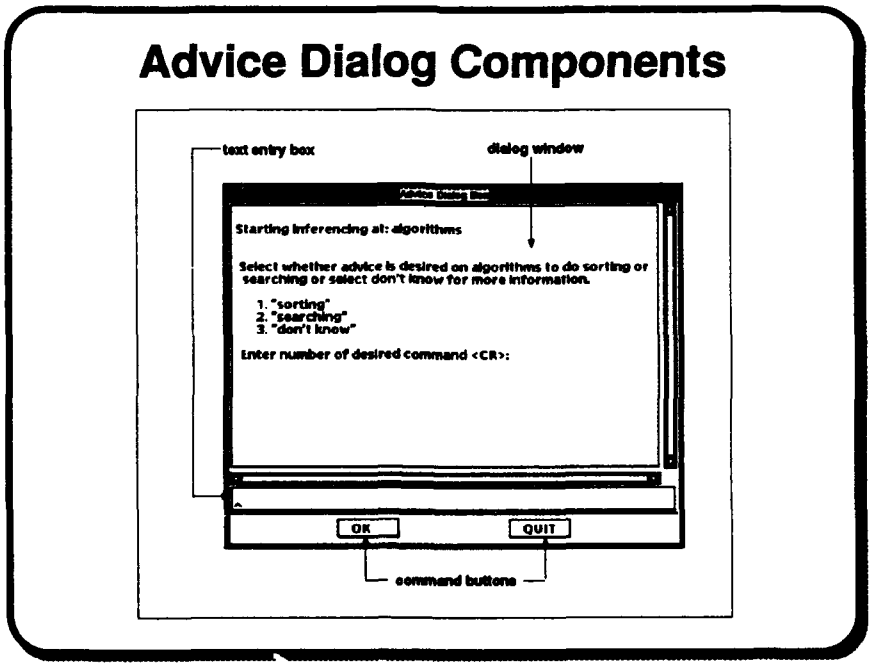
If no advice is available, the **Advice** option appears greyed out in the node Pop-Up menu. Otherwise, after selecting this option, the advisor dialogue is conducted in a Pop-Up window. At the beginning of an inference session, the user can control how the advice is presented and how much explanation is provided. Personalized inferencing preferences can be set via through entries in the `.rlfrc` file.

Slide 37



Inferencing is conducted through the use of a dialog box. The properties of this dialog box are shown on SLIDE 38.

Slide 38



As the user interacts with the advice mode of the librarian application, the user may be relocated to other nodes within the model. Depending on how the inferencing portion of the model is written, additional questions may be asked of the user and further movement in the model may take place. SLIDE 39 is a snapshot of a portion of an ongoing user interaction. Note that the user may end the inferencing session at any time by clicking on the **[Quit]**

THIS
PAGE
IS
MISSING
IN
ORIGINAL
DOCUMENT

30 + 31

Slide 42

RLF Usage

- Follow RLF Installation Instructions
- Provide access to RLF executables and scripts

  e.g. with UNIX command

  set path = (*RLF_GB_pathname* $path)
- Set RLF Environment variables; e.g.

  DISPLAY e.g., with UNIX command

  setenv DISPLAY *hostname*:0

  RLF_LIBRARIES e.g., with UNIX command

  setenv RLF_LIBRARIES *rlf_library_pathname*
- Invoke RLF, typically through the supplied script **RLF_GB**

Some of the options above can be permanently set by configuring the users `.cshrc` file. The **RLF_GB** script tracks the users UNIX environment for the necessary RLF-specific features and warns if they are not present or incorrect. The text of this script can be modified to suit individual user profiles.

Clearly there are many ways to use the RLF. The browser supports four distinct usage modes that can be used together to learn about and retrieve reusable assets. These are summarized on the next slide.

Slide 43

RLF Usage Modes

Browse -- User navigates the library directly using graphical interface

Advise -- User invokes **Advice** option to get rule-based guidance about categories and objects in the model

Query -- User invokes Search Command to locate Objects or Categories by name using a partial substring of the name

Action -- User invokes actions through which objects and categories can be inspected, analyzed and retrieved

There are many scenarios that interweave these modes to productively use an RLF library. Some specific scenarios are given in the RLF User Manual. To learn to use the `RLF GB`, the user should spend some time using the interface on the sample library models provided with the RLF installation. Some suggested practice "exercises" are given on the following slide.

Slide 44

RLF Learning Scenarios

- find a QuickSort routine in the sample "Sort and Search" library graphically and view it
- find a QuickSort routine in the sample "Sort and Search" library using the **Search** command and extract it
- investigate available sorting routines using the **Advice** command at the **`algorithms`** node -- see if you can locate the same QuickSort routine
- investigate the Search sub-model both graphically and textually -- how many node names contain the substring "ear"

The RLF is a software product that has grown from an initial proof-of-concept prototype to a large and fairly complex system that works in concert with the user's own computing environment in general, and the available X Window system in particular. Depending on

this computing environment, the availability of system resources, and the definition of library model definitions, an RLF application may report various errors to the user. The RLF User's manual discusses these errors in some detail and breaks them down in three categories:

- X Window System Errors
- RLF Errors
- File Processing Errors

The next three slides identify some general error messages and some possible novice user errors that may lead to them.

Slide 45

X Errors and Possible Causes

Error Message:

```
** MAIN PROGRAM ABANDONED -- EXCEPTION "constraint_error"
RAISED
```

May be caused by a X display setting problem -- check value of `DISPLAY` environment variable

Error Message:

```
Unexpected exit from browser. Unhandled event: <event_type>
```

This error signals a mis-communication between the application and the X server. If it persists, it can signal an X server/application version incompatibility.

Error Message:

```
X error: failed request ...
Bad value: integer value out of range ...
```

Error messages similar to this indicate incompatibility between the RLF GB application and the window system.

In any case, using an unmodified MIT X release (X11R4 or X11R5) is recommended. Both X-related errors and internal RLF errors are reported through the Ada exception mechanism. Sometimes the error message documents the Ada effect of the actual error rather than the error itself.

Slide 46

RLF Errors

Error Message:

```
** MAIN PROGRAM ABANDONED -- EXCEPTION "NAME_ERROR" RAISED
```

The exception raised can also be "FILE_NAME_ERROR" or "END_ERROR". This error is typically due to an improper RLF_LIBRARIES environment variable, or a corrupted Libraries directory.

Error Message:

```
** MAIN PROGRAM ABANDONED -- EXCEPTION "UNINITIALIZED_UID"
RAISED
```

This error signals that an RLF library has been left in a locked state. Check for files of the form *.LOK in the $RLF_LIBRARIES. directory and remove them.

Error Message:

```
** MAIN PROGRAM ABANDONED -- EXCEPTION
"UNINITIALIZED_OBJECT" RAISED
```

This error indicates corrupted RLF library data, possibly due to a system crash.

In the case of the last error on the previous slide, rebuilding the library model may be necessary to recover from the error.

Both RLF knowledge bases and the contents of asset objects are stored in files. For the former, RLF routines process the data and may report errors in the face of missing or corrupted files. The latter are processed by RLF action routines which may have their own unique way for detecting and handling file processing errors.

Slide 47

File Processing Errors

Library data is typically stored as UNIX files. These files are located within the **$RLF_LIBRARIES** directory. RLF's own knowledge-base files are located there and any errors are flagged by RLF error messages similar to those given on the previous slide.

Library data files themselves are typically processed through RLF actions. The action statements within the model refer to either internal procedures or external routines to process the files. Any errors that occur here will be specific to such action routines.

Library data files are usually located in **$RLF_LIBRARIES/Text** or (preferably) **$RLF_LIBRARIES/Text/*model_name***

6 Tailoring the RLF GB

While the RLF is quite usable with the default set-up provided by a standard RLF installation, there are many opportunities to customize it for appearance and for feature defaults, availability and presentation. This section surveys some of these customization options and SLIDE 48 lists the general customization areas available to the user.

Slide 48

RLF Customization Areas

- X Windows Resources
- Environment Variables
- Command Line options
- RLF Initialization file `.rlfrc`

Modifying X resources is generally beyond the scope of this orientation. X applications can have their resources customized at run-time through various means including entries in an application-specific file in `/usr/lib/X11/app-defaults`, through the X resource directory path named in the `XAPPLRESDIR` environment variable, through user-specific resource file entries such as those in `.Xdefaults`, or on the command line. The most convenient method for tailoring such resources for the RLF is to copy an `RLF_Browser` file to a convenient location, modify it as appropriate, and use the value of `XAPPLRESDIR` to refer to this file.

In addition, the bitmaps and fonts for the **RLF GB** application can be changed. Making these changes is discussed in the RLF User's Manual.

Two RLF-specific environment variables have been mentioned previously. The RLF also uses the environment variables `RLF_PAGER` and `RLF_EDITOR` to indicate which default text viewer and editor to use in RLF actions which process text files. The defaults are *less*, a public domain text browsing program similar to *more*, and *vi*. The environment variables can be set to override these defaults; e.g. `setenv RLF_EDITOR emacs`.

The effect of RLF command line options can be learned by typing `Graphical_Browser -help`. The produces the output shown on the next slide.

Slide 49

RLF Command Line Options

```
Welcome to the RLF Graphical Browser.
          Version 4.1

Copyright 1993, Paramax Systems Corp.


Available command line arguments:

   -help          prints available command line arguments
   -I <pname>     uses RLF libraries found in the directory
                  specified by pathname <pname>
   -l <name>      specifies name of the library to browse
   -e <fname>     specifies filename of editor to use to edit
                  text associated to a network
   -p <fname>     specifies filename of pager used to view
                  text associated to a network
   -d             enables debug messages from the RLF tools
```

Perhaps the most convenient point for users to provide customization of the RLF is by modifying the `.rlfrc` file. This file is easy to read and understand – the full BNF specification is given in the RLF User's Manual. SLIDE 50 summarizes the options which can be set with this file.

Slide 50

RLF Initialization Options

- Library Instances Directory
- Default Library
- Initial Category
- View Control
- Bitmap changes
- Advice Control
- Translator Settings

In general, following UNIX conventions, conflicting values established on an RLF command line override those set earlier (either as environment variables or as `.rlfrc` values, while any surviving environment variables, override any values set in the `.rlfrc` file. Thus, if the `RLF_LIBRARIES` environment variable is set, it will override any `library directory` entry in the `.rlfrc` file.

A default library setting will cause the browser to open that library immediately. Within a library, a specific node can be chosen as the first focal point of the browser. View control can be used to select whether the topology view is on or off at start-up, and whether the first graphical view drawn is a specialization view or relationship view. The depth of the view can also be set.

If a user does not specify any advice options, the user must answer some start-up questions to select preferences during the first inferencing session. If a user wishes to establish alternate bitmaps for model icons, these can be set-up through the `.rlfrc` as well. The user can also select some options to be observed when running the RLF language translators **Lmdl** and **Rbdl**.